Timing Problem for Scheduling an Airbone Radar

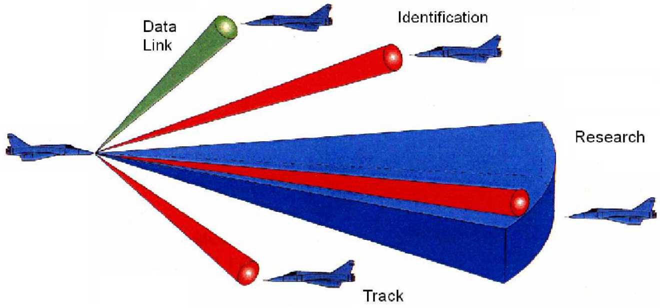
Yann Hendel **Ruslan Sadykov**

LIX, Ecole Polytechnique, France

PMS'08, Istanbul
April 29, 2008

Contents

Scheduling an airbone radar



- Several cyclic jobs
- A modern radar
- We want the jobs frequency to be close to desired as much as possible

Model I

Due to [Winter, Baptiste, 07]

Data

- Time horizon $H = \{0, \ldots, h-1\}$
- Set of jobs $N = \{1, \ldots, n\}$
- For each job $i \in N$:
  - Chain of operaions $(O_{i0}, O_{i1}, \ldots, O_{i,n(i)})$
  - Starting time S_{i0} of operation O_{i0}
  - Processing time p_i of an operation
  - Desired time l_i between two operations
  - Penalty function $\delta_i(x) = \max\{\alpha_i(l_i - x), \beta_i(x - l_i)\}$

Variables

S_{ij} — starting time of operation O_{ij}, $i \in N$, $j \in N(i)$

Model II

Objective function

Find a feasible schedule $\{S_{ij}\}_{i \in N, j \in N(i)}$ which minimizes the total penalty

$$F = \sum_{i \in N} \sum_{j \in N(i)} \delta_i (S_{ij} - S_{i,j-1})$$

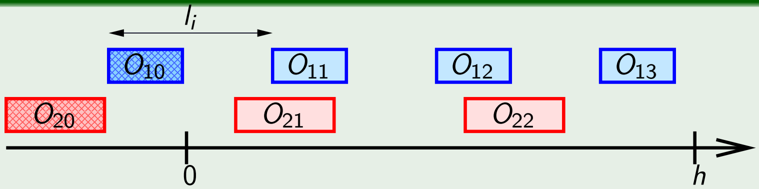
Model II

Objective function

Find a feasible schedule $\{S_{ij}\}_{i\in N, j\in N(i)}$ which minimizes the total penalty

$$F = \sum_{i\in N} \sum_{j\in N(i)} \delta_i(S_{ij} - S_{i,j-1})$$

Example



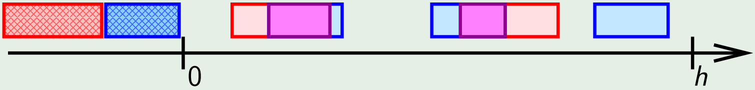
Model II

Objective function

Find a feasible schedule $\{S_{ij}\}_{i \in N, j \in N(i)}$ which minimizes the total penalty

$$F = \sum_{i \in N} \sum_{j \in N(i)} \delta_i (S_{ij} - S_{i,j-1})$$

Example



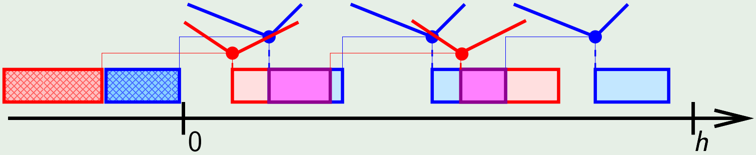
Model II

Objective function

Find a feasible schedule $\{S_{ij}\}_{i \in N, j \in N(i)}$ which minimizes the total penalty

$$F = \sum_{i \in N} \sum_{j \in N(i)} \delta_i(S_{ij} - S_{i,j-1})$$

Example



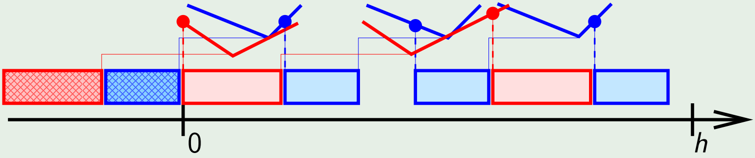
Model II

Objective function

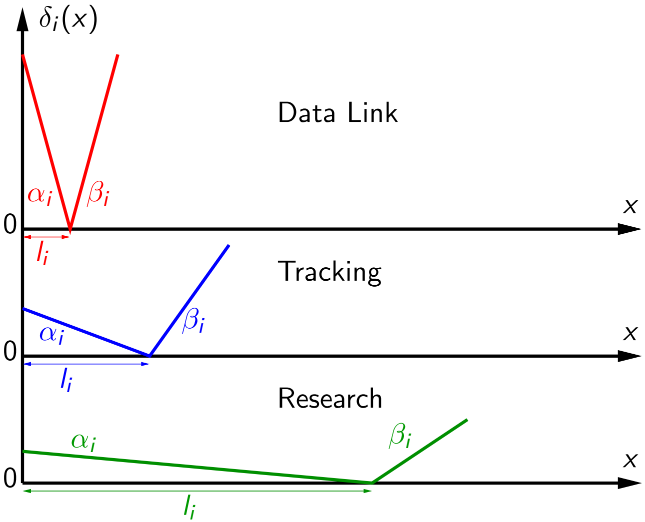
Find a feasible schedule $\{S_{ij}\}_{i \in N, j \in N(i)}$ which minimizes the total penalty

$$F = \sum_{i \in N} \sum_{j \in N(i)} \delta_i(S_{ij} - S_{i,j-1})$$

Example



Examples of penalty functions



Relations with other problems

- Generalization of the just-in-time scheduling problem
  $1 \mid\mid \sum E_j + T_j$
  (hence NP-complete [Garey, Tarjan, Wilfong, 88])
- The cyclic version of the problem is a generalization of
  - the non-preemptive scheduling for Distance-Constrained Task System [Han, Lin, 92]
  - the Periodic Maintenance Problem [Wei, Liu, 83]

Relations with other problems

- Generalization of the just-in-time scheduling problem
  $1 \mid\mid \sum E_j + T_j$
  (hence NP-complete [Garey, Tarjan, Wilfong, 88])
- The cyclic version of the problem is a generalization of
  - the non-preemptive scheduling for
    Distance-Constrained Task System [Han, Lin, 92]
  - the Periodic Maintenance Problem [Wei, Liu, 83]

The local search : why and how ?

Practical requirement

We need to give a solution in 1 second !

- Local search seems to be the best we can do
- Neighborhoods are determined by sequences of jobs
- The timing subproblem : given a sequence of jobs, find a schedule which minimizes the total penalty
  - the key problem for local search algorithms
  - not trivial in a just-in-time environment

The local search : why and how ?

Practical requirement

We need to give a solution in 1 second !

- Local search seems to be the best we can do
- Neighborhoods are determined by sequences of jobs
- The timing subproblem : given a sequence of jobs, find a schedule which minimizes the total penalty
  - the key problem for local search algorithms
  - not trivial in a just-in-time environment

The local search : why and how ?

Practical requirement

We need to give a solution in 1 second !

- Local search seems to be the best we can do
- Neighborhoods are determined by sequences of jobs
- The timing subproblem : given a sequence of jobs, find a schedule which minimizes the total penalty
  - the key problem for local search algorithms
  - not trivial in a just-in-time environment

The local search : why and how ?

Practical requirement

We need to give a solution in 1 second !

- Local search seems to be the best we can do
- Neighborhoods are determined by sequences of jobs
- The timing subproblem : given a sequence of jobs, find a schedule which minimizes the total penalty
  - the key problem for local search algorithms
  - not trivial in a just-in-time environment

LP Formulation

- Data :
  - A sequence of operations $K = \{1, \ldots, k\}$
  - $b(j)$ — the "reference" operation for operation j
  - d_j — desired distance between operations j and $b(j)$
- Variables :
  - S_j — the starting time of operation j
  - E_j, T_j — the earliness and tardiness of operation j

LP Formulation

- Data :
  - A sequence of operations $K = \{1, \ldots, k\}$
  - $b(j)$ — the “reference” operation for operation j
  - d_j — desired distance between operations j and $b(j)$
- Variables :
  - S_j — the starting time of operation j
  - E_j, T_j — the earliness and tardiness of operation j

LP Formulation

- Data :
  - A sequence of operations $K = \{1, \dots, k\}$
  - $b(j)$ — the "reference" operation for operation j
  - d_j — desired distance between operations j and $b(j)$
- Variables :
  - S_j — the starting time of operation j
  - E_j, T_j — the earliness and tardiness of operation j

$$
\min \sum_{j \in K} \alpha_j E_j + \sum_{j \in K} \beta_j T_j
$$

$$
s.t. \left\{ \begin{array}{rcll}
S_j - S_{j-1} & \geq & p_{j-1}, & j \in K, \\
E_j & \geq & d_j - (S_j - S_{b(j)}), & j \in K, \\
T_j & \geq & S_j - S_{b(j)} - d_j, & j \in K, \\
 & & T_j \geq 0, \quad E_j \geq 0, & j \in K.
\end{array} \right.
$$

LP Formulation

- Data :
  - A sequence of operations $K = \{1, \ldots, k\}$
  - $b(j)$ — the "reference" operation for operation j
  - d_j — desired distance between operations j and $b(j)$
- Variables :
  - S_j — the starting time of operation j
  - E_j, T_j — the earliness and tardiness of operation j

$$\max \quad -\sum_{j \in K} \alpha_j E_j - \sum_{j \in K} \beta_j T_j$$

$$s.t. \quad \left\{ \begin{array}{lcllr} -S_j + S_{j-1} & \leq & -p_{j-1}, & j \in K, & (F_{j-1 \to j}) \\ -S_j + S_{b(j)} - E_j & \leq & -d_j, & j \in K, & (F_{j \to b(j)}) \\ -S_{b(j)} + S_j - T_j & \leq & d_j, & j \in K, & (F_{b(j) \to j}) \\ T_j \geq 0, \quad E_j \geq 0, & & & j \in K. & \end{array} \right.$$

Dual (MinCostFlow) Formulation

- Data :
  - A sequence of operations $K = \{1, \ldots, k\}$
  - $b(j)$ — the "reference" operation for operation j
  - d_j — desired distance between operations j and $b(j)$
- Variables :
  - $F_{i\to j}$ — the flow from operation i to operation j

$$\min \quad \sum_{j\in K} d_j F_{b(j)\to j} - \sum_{j\in K} d_j F_{j\to b(j)} - \sum_{j\in K} p_j F_{j\to j-1}$$

$$s.t. \begin{cases} \displaystyle\sum_{i\in\{j+1,b(j)\}} F_{j\to i} - \sum_{i\in\{j-1\}\cup\{i':b(i')=j\}} F_{i\to j} = 0, & j \in K \cup \{0\}, \\ 0 \le F_{j\to j-1}, & j \in K, \\ 0 \le F_{j\to b(j)} \le \alpha_j, & j \in K, \\ 0 \le F_{b(j)\to j} \le \beta_j, & j \in K. \end{cases}$$

Example

Data ($n = 3$, $h = 50$)

$n(i)$	p_i	l_i	(α_i, β_i)	S_{i0}
3	6	17	(1,1)	-6
2	5	25	(1,1)	-11
1	4	50	(1,1)	-15

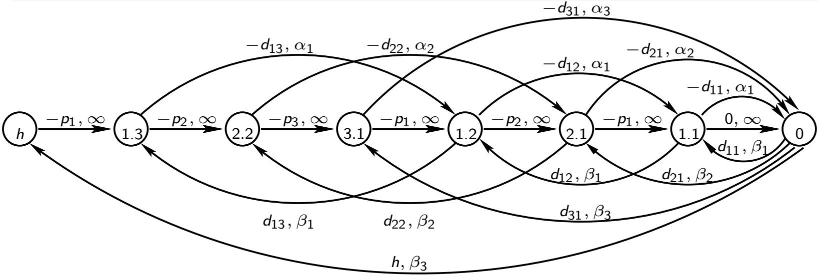
Sequence : $O_{11} \rightarrow O_{21} \rightarrow O_{12} \rightarrow O_{31} \rightarrow O_{22} \rightarrow O_{13}$

Example

Data ($n = 3,\ h = 50$)

$n(i)$	p_i	l_i	(α_i, β_i)	S_{i0}
3	6	17	(1,1)	-6
2	5	25	(1,1)	-11
1	4	50	(1,1)	-15

Sequence : $O_{11} \rightarrow O_{21} \rightarrow O_{12} \rightarrow O_{31} \rightarrow O_{22} \rightarrow O_{13}$

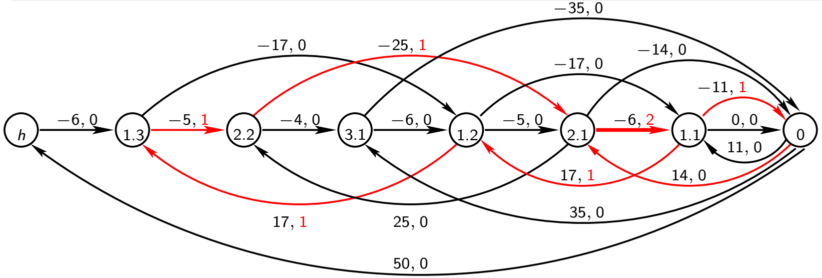


Example

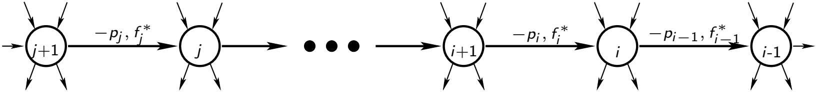
Data ($n = 3,\ h = 50$)

$n(i)$	p_i	l_i	(α_i, β_i)	S_{i0}
3	6	17	(1,1)	-6
2	5	25	(1,1)	-11
1	4	50	(1,1)	-15

Sequence : $O_{11} \rightarrow O_{21} \rightarrow O_{12} \rightarrow O_{31} \rightarrow O_{22} \rightarrow O_{13}$

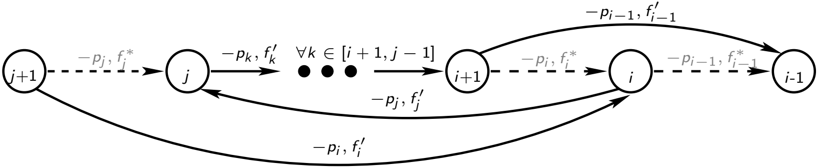


Detecting worse sequences in a neighborhood



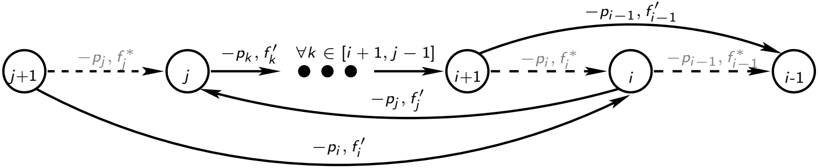
Detecting worse sequences in a neighborhood

We insert operation i just after operation j



Detecting worse sequences in a neighborhood

We insert operation i just after operation j

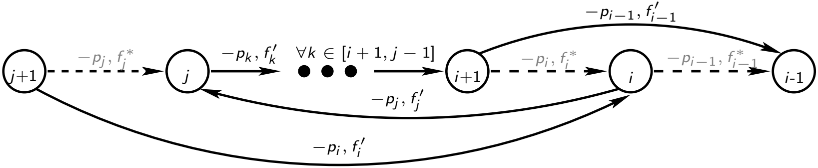


New flow f' is feasible if

- $f_i' = f_j^*$
- $f_{i-1}' = f_{i-1}^*$
- $f_j' = f_j^* - f_i^* + f_{i-1}^* \geq 0$
- $f_k' = f_k^* + f_{i-1}^* - f_i^* \geq 0, \forall k \in [i+1, j-1]$

Detecting worse sequences in a neighborhood

We insert operation i just after operation j



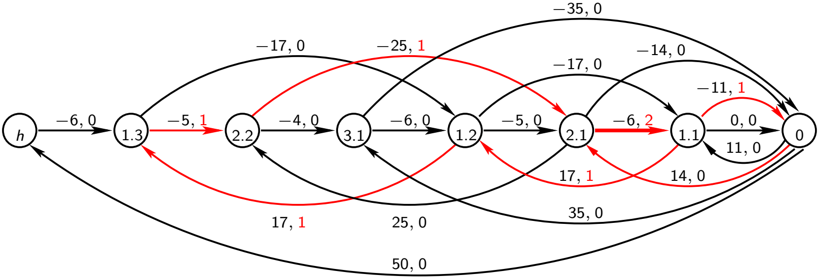
New flow f' is feasible if

- $f'_i = f^*_j$
- $f'_{i-1} = f^*_{i-1}$
- $f'_j = f^*_j - f^*_i + f^*_{i-1} \geq 0$
- $f'_k = f^*_k + f^*_{i-1} - f^*_i \geq 0, \forall k \in [i+1, j-1]$

$$\text{if} \quad f^*_i \left(p_j + p_i + \sum_{k=i+1}^{j-1} p_k \right) \leq f^*_j p_i + f^*_{i-1} \left(p_j + \sum_{k=i+1}^{j-1} p_k \right)$$

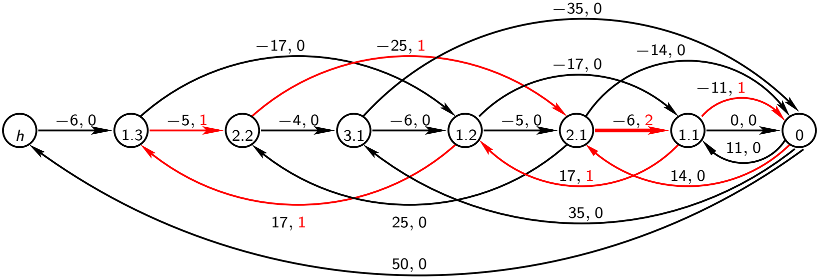
then $\mathrm{cost}(f') \leq \mathrm{cost}(f^*)$ (insertion will not give a better solution)

Example (continued)



- 12 insertions are possible in total,
- 9 of them will not allow us to improve the current solution

Example (continued)



- 12 insertions are possible in total,
- 9 of them will not allow us to improve the current solution

Preliminary numerical experiments

- "Best fit" local search with the Insertion neighborhood
- Instances designed together with radars specialists
- MinCostFlow solver : MCFZIB
  (≈20 times faster than the Cplex LP solver)
- We devised exact algorithms for the problem [Baptiste, S., 08]

Preliminary numerical experiments

- “Best fit” local search with the Insertion neighborhood
- Instances designed together with radars specialists
- MinCostFlow solver : MCFZIB
  ($\approx$20 times faster than the Cplex LP solver)
- We devised exact algorithms for the problem [Baptiste, S., 08]

Preliminary numerical experiments

- "Best fit" local search with the Insertion neighborhood
- Instances designed together with radars specialists
- MinCostFlow solver : MCFZIB
  ($\approx$20 times faster than the Cplex LP solver)
- We devised exact algorithms for the problem [Baptiste, S., 08]

n	$\sum n(i)$	Load	Gap	Time	# of timings	Reduction
9	57	67%	0%	0.02s	15	99%
14	62	70%	0%	0.04s	66	93%
	83	99%	0.6%	0.42s	756	95%
15	91	85%	38.9%	4.16s	6260	75%
	93	90%	>100%	4.80s	6820	75%
18	72	85%	1.4%	0.25s	702	83%
	74	91%	1.3%	0.55s	1341	76%
	76	97%	1.1%	0.63s	1430	84%

Conclusions

- A real-life problem
- Relatively fast way to solve the timing subproblem
- A way to speed-up the local search withour loss of its quality
- Good results for a majority of practical instances
- Work in progress to improve results